BugReports for TeamD

General methods used in testing

- We mainly focused on black box testing, where we manual tested the applications via command line input. The focuses of our black box testing are:
    - Normal values
    - Exception-causing values
    - Boundary values (e.g.replacement index -1, 0, 1 for sed)
    - Combination of cases (e.g. for pipe command A| B, where A and B can be a valid command, an invalid command, a command that succeeds, command that throws an exception, command that are applicable/not applicable for piping, etc)

- White box testing is conducted by reusing unit/integration test cases. The focuses of the white box testing are:
    - Test for conditions that cannot be tested through manual testing (eg null input stream, specific function implementations, etc).
    - Attempt to discover any error and faults that do not manifest as failures during black box testing via command line.
    - For the test cases provided, change the package name to according to your team's requirements.

- Random testing
    - Test random and uncommon scenarios (e.g. cd ../../<currentDirectory>)

Test results

51 bugs discovered

Bug reports table

Bug Report Number	Description	Testcase	Comments (this column should be left empty; for marking usage)
1	. is not treated as a directory, happens on multiple apps Expected: . should be treated as a directory, i.e. ls . should print contents of current directory or cat . should print “error this is a directory” Actual: Output for all applications is null	ls . cat . mkdir .	
2	Globbing does not work, * is not parsed correctly Expected: (See project requirements) For each argument ARG in a shell command that contains unquoted * (asterisk) do the following: 1. Collect all the paths to existing files and directories such that these paths can be obtained by replacing all the unquoted asterisk symbols in ARG by some (possibly empty) sequences of non-slash characters. 2. If there are no such paths, leave ARG without changes. 3. If there are such paths, replace ARG with a list of these path separated by spaces. Actual: Output for all applications are null	ls * cat *	

3	Special characters (%, &, / etc) are not parsed correctly Expected: They should be treated as invalid arguments Actual: Output for all applications are null	ls / cat % mkdir ^	
4	Numbers are not parsed correctly Expected: They should be treated as file/directory names (string) and applications should be executed accordingly. Actual: Output for all applications are null	ls 1 cd 0	
5	Multiple spaces after and app command causes parsing error Expected: <app_command> <any number of spaces> should treat the spaces as an empty argument, i.e. same as <app_command> Actual: <app_command> <any number of spaces> throws ShellException for Invalid Syntax for all applications	ls <space> wc <space>	

6	Empty quotes “” are not handled Expected: Empty quotes should be treated as empty arguments and the application should be executed accordingly. i.e. ls “” should behave the same as ls and print contents of current directory Actual: For most applications (except cat, echo), they print “String index out of range: 0”	find “” ls “”	
7	I/O redirection with ‘<’ and ‘>’, it does not work properly for all apps that allows input/output redirection. - When read from an existing/non-existing file, the app will crash - When write to a non-existing file, the new file with a wrong filename is created (should create ‘test.txt’ but create ‘est.txt’ instead) Possible cause: parse filename wrongly	Input example: grep “t” < testGrep.txt → app crashes Output example: grep “t” testGrep.txt > hello.txt → creates ello.txt	
8	Pipe does not work correctly. The behaviour varies for different applications, but an error will be thrown. *Example 1:* cat testGrep.txt | grep "t" Expected: Should return This is a test for grep application of the Shell APPLICATION that we implemented!	See description	

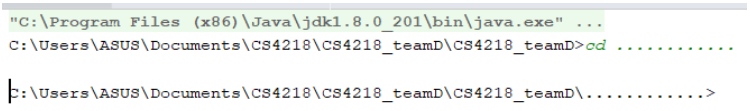
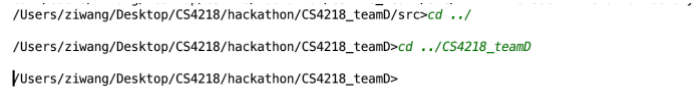
	Actual:
	Currently, error is thrown.
	cat: Could not read file
	
	Example 2: sort test.txt | sed "s/e/o/"
	
	Expected:
	Should print the sed result of the sorted content of test.txt
	
	Actual:
	Throw exception with empty exception message
	`/Users/ziwang/Desktop/CS4218/hackathon/CS4218_teamD>sort test.txt | sed "s/e/o/"`
	`sort:`
	`/Users/ziwang/Desktop/CS4218/hackathon/CS4218_teamD>`
	
	Example 3: echo “Hello” | grep “H”
	
	Expected:
	Should return Hello
	
	Actual:
	It returns
	Hello grep H
	java.io.PrintStream cannot be cast to
	java.io.ByteArrayOutputStream		

9	Command substitution is not working fully as intended. It should concatenate the output to the previous argument (as stated in ArgumentResolver.java) Expected: echo test`echo Test` should produce testTest Actual:	echo test`echo Test`	

	Currently, it returns with a space, test Test		
10	Command substitution for echo fails when multiple arguments are present Expected: echo `echo 1 2 3` should produce 1 2 3 Actual: It returns null	echo `echo 1 2 3`	
11	Echo: Echo should print an empty string if given zero arguments Expected: Should produce a new line Actual: It returns echo: Empty arguments	echo Refer to unit test, EchoApplicationTest.java runZeroArgumentsStdInputSuccess()	
12	Echo: Echo should throw an exception if given a null outputStream Expected: Should throw an exception Actual: Nothing is thrown	Refer to unit test EchoApplicationTest.java, runNullOutputThrowException()	
13	Wc: Wc does not consider \n and \r characters when counting bytes. Wc should consider only \n for counting number of lines (as per specified in Project Description) Expected: 4 12 60 testWc.txt	wc testWc.txt testWc.txt Hello\r\n (5+2) This should have\r\n (16+2) 4 lines and\r\n (11+2) 12 words and\r\n (12+2)	

	Actual: 5 12 52 testWc.txt	60 bytes (8)	
14	Ls: Expected: ls .. lists contents of parent folder Actual: Regardless of current directory, ls .. only prints “CS4218_teamD”	ls ..	
15	Ls: When in /CS4218_teamD/src, do ls src. Expected: Should print an error message saying src does not exist, as src is not a valid directory in src Actual: Prints contents of src folder.	cd src ls src	
16	Ls with Globbing: Expected: Should standardize and list either all relative paths or all absolute paths for results Actual: Lists relative paths for results with no globbing, and lists	ls src/META-INF src/sg vs ls src/*	

	absolute path for results with globbing C:\Users\ASUS\Documents\CS4218\CS4218_teamD\CS4218_teamD>*ls src/** C:\Users\ASUS\Documents\CS4218\CS4218_teamD\CS4218_teamD\src\META-INF: MANIFEST.MF C:\Users\ASUS\Documents\CS4218\CS4218_teamD\CS4218_teamD\src\sg: edu C:\Users\ASUS\Documents\CS4218\CS4218_teamD\CS4218_teamD>*ls src/META-INF src/sg* src/META-INF: MANIFEST.MF src/sg: edu C:\Users\ASUS\Documents\CS4218\CS4218_teamD\CS4218_teamD>		
17	Cd: Expected: cd a b c prints exception message that says “too many arguments” Actual: cd a b c prints null	cd a b c	
18	Cd: Expected: cd a prints an exception message that says “folder does not exist” Actual: cd a prints null	cd a	
19	Cd: Expected: cd …… is an invalid directory, should show an error message	cd ……...	

	Actual: User is able to cd into an invalid directory. Environment.currentDirectory is updated. 		
20	Cd: Any cd command will produce an extra new empty line. Not consistent with the other applications. 	Any cd command	
21	Cat: Expected: cat a prints an exception message that says “file does not exist” Actual: cat a prints null.	cat a	
22	Cat: Place testGrep.txt and testGrep2.txt in the same base folder as where test directory is found (ie base folder where src is also found). It should be able to print the contents of testGrep.txt and testGrep2.txt after printing that test is a directory. Expected: cat: This is a directory	cat test*	

	This is a test for grep application of the Shell APPLICATION that we implemented! Of course~ Hello I am grep. Grep this grep that i am cool like This and THAT Actual: cat: This is a directory		
23	Cat: Expected: Should throw error 'No such file or directory'. However, if there is such a filename, then it should print the contents of that file. Actual: Illegal char <*> at index 4: test*a	cat test*a	
24	Cat: Null InputStream and null arguments not handled. Expected: Should throw CatException Actual: throws NullPointerException	Refer to unit test CatApplicationTest.java, runNullInputStreamNullArgsFail()	

25	Mkdir: Current working directory is root folder. Expected: When src already exists in the root folder, mkdir src should throw an exception saying “could not create directory” Actual: it does not throw an exception	mkdir src mkdir src newFolder	
26	Mkdir: mkdir creates the directory in the wrong folder. In src, do mkdir hello Expected: A new directory called “hello” is created in /src, along with /sg and /META-INF Actual: A new directory is created in the root folder.	cd src mkdir hello	
27	Find: Expected: Should return bad option/flag. Actual: Currently, it throws null	find testFind -	
28	Find: Wildcard * for filename does not work. Should return all the files and folders that match the regex, ie	find testFind -name "L*"	

	starts with ‘L’. Expected: testFind\L1 testFind\L1\L1.txt testFind\L2 testFind\L2\L2 testFind\L2\L2\L2.txt testFind\L3 testFind\L4 testFind\L1.txt testFind\L2.txt Actual: Currently, nothing is returned.	Refer to unit test FindApplicationTest.java, runFindOneFolderWildcardFilenameSpecifiedSuccess()	
29	Find: Wildcard * for filename does not work. Should return all the files and folders that match the regex, ie ends with ‘.txt’. Expected: testFind\L1\File1.txt testFind\L1\L1.txt testFind\L2\File2\file2.txt testFind\L2\L2\File2.txt testFind\L2\L2\L2.txt testFind\L2\test.txt testFind\L1.txt testFind\L2.txt Actual: When run from Shell, invalid error syntax is thrown. find: Invalid syntax. Check that an argument is given after -name	find testFind -name "*.txt" Refer to unit test FindApplicationTest.java, runFindTwoFoldersWildcardFilenameSpecifiedSuccess()	

	When run from the unit test, PatternSyntaxException: Dangling meta character '*' near index 0 *.txt ^ is thrown.		
30	Find: Expected: Should return nothing as there is no such file name. Actual: Currently, it returns the directory L1. testFind\L1	find testFind -name "L1/*"	
31	Find: The correct file is returned but the file separator is inconsistent. Expected: testFind\L1\L1.txt Actual: testFind/L1\L1.txt	find testFind/L1 -name "L1.txt"	
32	Find: Current directory is [initial pwd path]/testFind/L2. Expected: Should not return results as there is no filename ‘L1.txt’ in folder testFind/L2. Actual: Currently, results are returned. .\testFind\L1.txt .\testFind\L1\L1.txt	After cd into testFind/L2, find . -name "L1.txt"	

33	Find: Current directory does not contain testFind. Expected: Results returned should contain the relative path from current working directory to the file. .\CS4218_teamD\testFind\L1.txt .\CS4218_teamD\testFind\L1\L1.txt Actual: Currently, the relative path is not displayed and it returns .\testFind\L1.txt .\testFind\L1\L1.txt	After one cd .. , where the current directory does not contain the folder testFind, find . -name "L1.txt"	
34	Find: cd one directory above and find the file given the relative path. Expected: Results returned should contain the relative path from current working directory to the file. CS4218_teamD\testFind\L1.txt CS4218_teamD\testFind\L1\L1.txt Actual: Currently, error is thrown. find: nonExistentFolder: No such file or directory	After one cd .. , give relative path from current directory to testFind, find CS4218_teamD\testFind -name "L1.txt"	
35	Find: Expected: Should return no folder specified. Actual: Currently, it throws find: Invalid syntax. Check that an argument is given after	find -name "L1.txt" Refer to unit test FindApplicationTest.java, runFindNoFolderSpecifiedThrowsFindException()	

	-name		
36	Grep: Expected: Should return some meaningful error (eg ‘Missing Arguments’). Actual: Currently, the application cannot terminate.	grep	
37	Grep: Expected: Should throw error ‘no such file’. Actual: Currently, it throws grep: IO not working	grep "t" nonexist.txt Refer to unit test GrepApplicationTest.java, grepFromFilesNonexistentFilePrintFileNotFoundSuccess()	
38	Grep: Expected: Should throw error that that it is a directory. Actual: Currently, it throws grep: IO not working	grep "t" src Refer to unit test GrepApplicationTest.java, grepFromFilesDirectoryPrintIsDirectorySuccess()	
39	Grep: Expected: It should either throw an error that the flag is invalid or take both ‘i’ and ‘c’ as parameters. If it takes both ‘ic’ as parameters, it should return 4	grep -ic "t" testGrep.txt	

	Actual: When 2 flags are used together, only the first flag (-i) is recognised and the results are shown based on that. This is a test for grep application of the Shell APPLICATION that we implemented!		
40	Grep: Expected: It should throw an error that the flag is invalid. Actual: Does not throw exception and results for flag -i is calculated. This is a test for grep application of the Shell APPLICATION that we implemented!	grep -in "t" testGrep.txt	
41	Grep: Correct results are returned but there should be clear indication which results belong to which file, similar to in UNIX. Expected: testGrep.txt: This is a test for testGrep2.txt: cool like This and THAT Actual: This is a test for cool like This and THAT	grep "This" testGrep.txt testGrep2.txt	
42	Grep:	grep testGrep.txt "is*"	

	Expected: Should return no such file. Actual: Currently, it throws Illegal char <*> at index 2: is*		
43	Grep: Expected: Should return no results because an empty pattern is passed. Actual: Currently, results are returned. This is a test for grep application of the Shell APPLICATION that we implemented! Of course~	grep "" testGrep.txt	
44	Grep: Expected: Should throw an error that no arguments supplied. Actual: Currently, the application cannot terminate.	grep -i Refer to unit test GrepApplicationTest.java, runNoPatternThrowsGrepException()	
45	Grep: When stdin is null and no input file is supplied, the correct error message should be missing input stream or file. It should not throw wrong number of arguments as the number of arguments can be correct if it is part of a pipe command, eg cat test.txt | grep “n$”	Refer to unit test GrepApplicationTest.java, runMissingInputStreamThrowsGrepException()	

	Expected: Should throw missing input stream or file. Actual: grep: Not correct number of arguments for grep!		

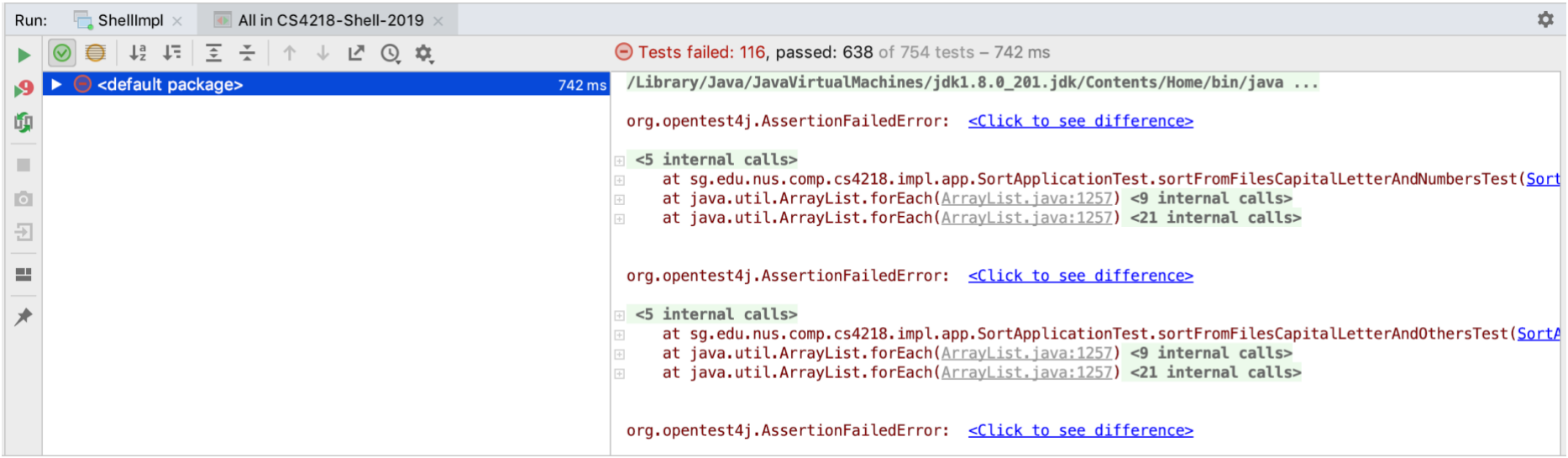
46	Sort:
	
	Exception message is not clear
	`/Users/ziwang/Desktop/CS4218/hackathon/CS4218_teamD>sort *n`
	`sort:`
	`/Users/ziwang/Desktop/CS4218/hackathon/CS4218_teamD>`
		sort *n test.txt	

47	Date: Run with “%%” as format Expected: Throw invalid format exception, or if completely following UNIX, print “%” as result Actual: Print “%%” as result	Refer to unit test DateApplicationTest.java, testFormatContainsDoublePercentShouldFail()	
48	Date: Run DateApplication.run() with null arguments Expected: Throw exception Actual: Does not throw exception	Refer to unit test DateApplicationTest.java, testNullDateFormat()	
49	Sed:	Refer to unit test	

	Run Sed with insufficient delimeters Expected: Throw exception Actual: Does not throw exception, print result as if there is enough delimeters	SedApplicationTest.java, runInsufficientDelimitersTest()	
50	Sed: Run Sed with extra delimeters Expected: Throw exception Actual: Does not throw exception, print result as if the extra delimiter is ignore	Refer to unit test SedApplicationTest.java, runExtraDelimitersTest()	
51	Sed: Run Sed with more than one file Expected: Print replaced result from all the files Actual: Throw exception “sed: stdin is missing.”	Refer to unit test SedApplicationTest.java, runMoreThanOneFileArgumentTest	

Other comments

- All of the 638 test cases provided by the team passed on Windows but 116 failed on Mac OS



~END~